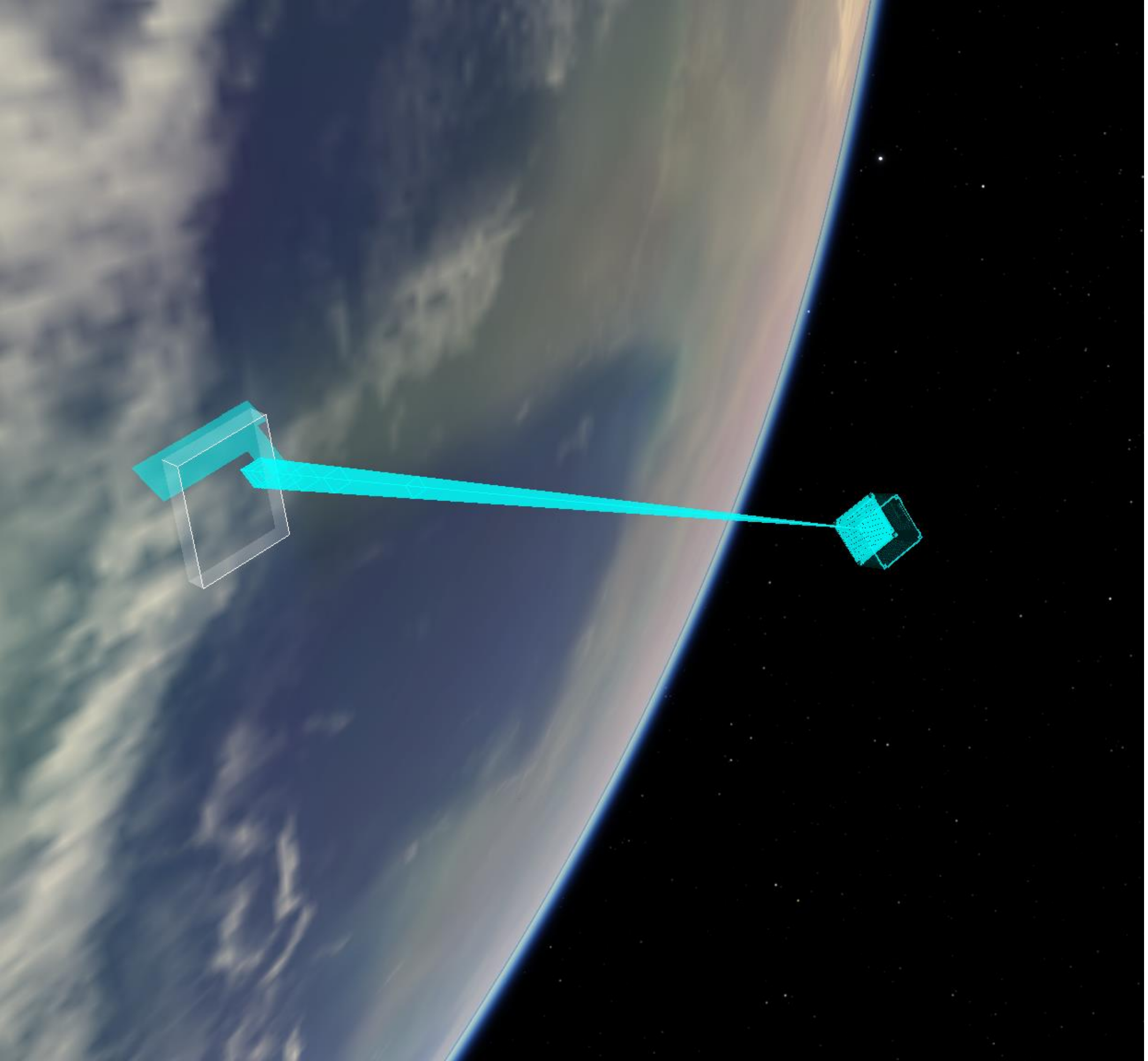


Scheduling and Planning Applications woRKshop (SPARK)

An ICAPS 2018 Workshop

Delft, The Netherlands

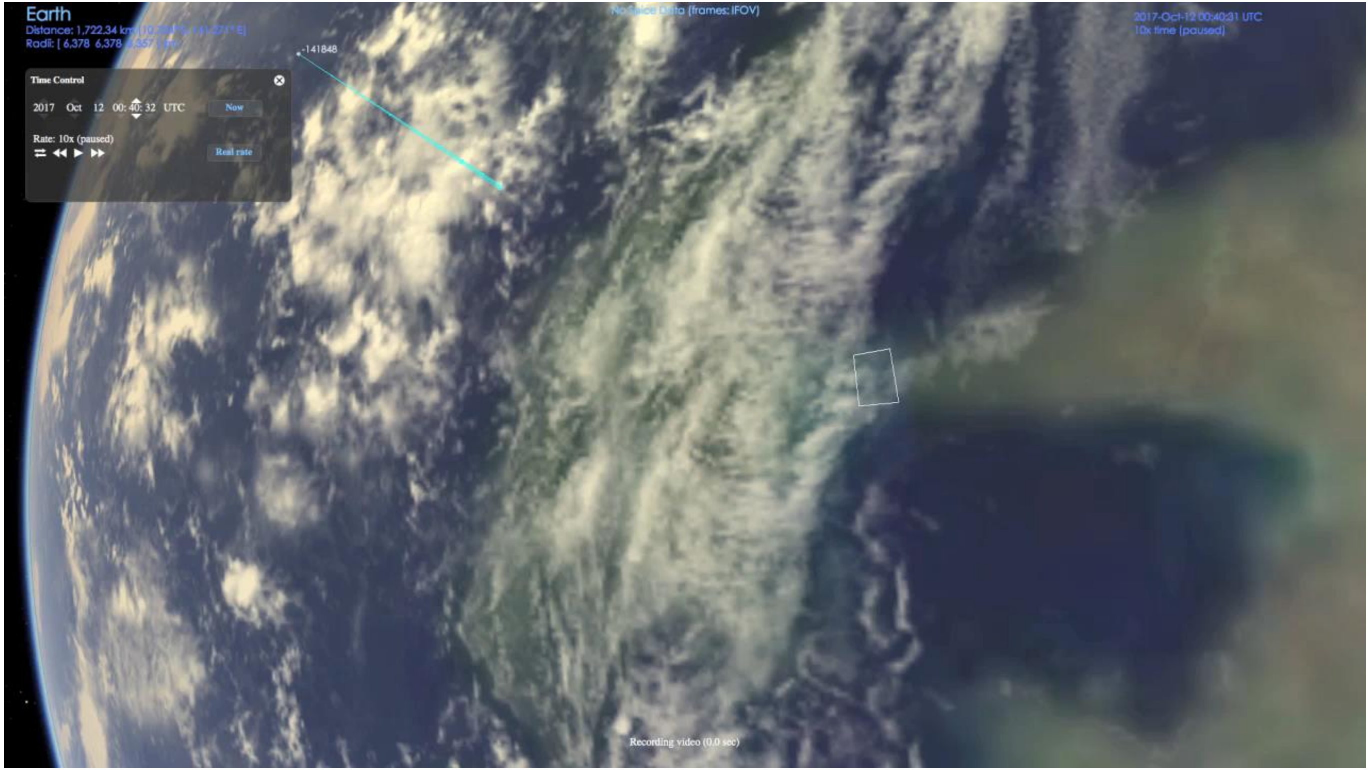
Area Coverage Planning with 3-axis Steerable, 2D Framing Sensors

Elly Shao, Amos Byon, Chris Davies, Evan Davis, Russell Knight, Garett Lewellen, Michael Trowbridge and Steve Chien

Jet Propulsion Laboratory, California Institute of Technology

Jet Propulsion Laboratory
California Institute of Technology

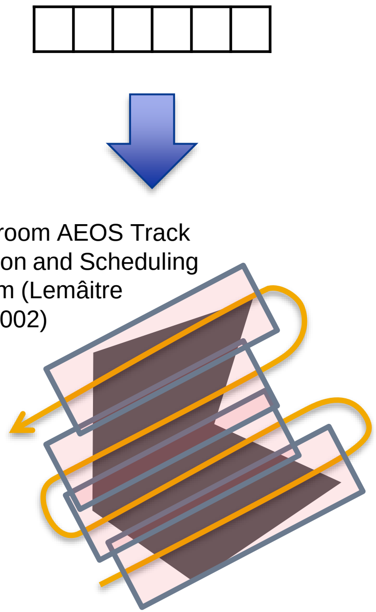
Example of Area Coverage with a 2D Framing Sensor



State of the Art

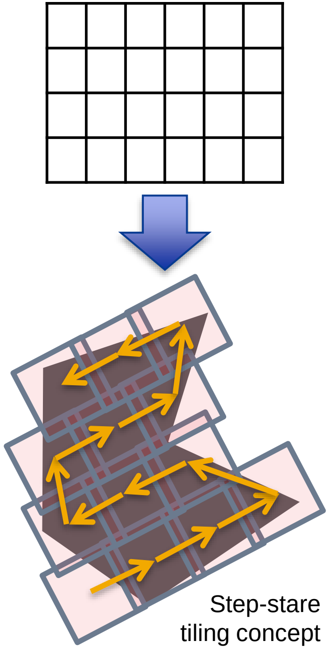
Prior work	Sensor type	Planning Approach
Agile Earth Observing Satellite (PLEIADES) scheduling (Lemâitre et al. 2002)	Pushbroom	Strip-based Boustrophedon decomposition (Choset and Pignon 1998).
Eagle Eye ISS Telescope (Knight, Donellan and Green 2013) (proposed mission)	Framing	Points only. No area algorithms published.
Planet Labs Flock (Boshuizen et. al 2014)	Framing	Don't plan individual targets. Launch many CubeSats and image whole Earth continuously at nadir, 1 Hz.
Mission to Understand Ice Retreat (Knight, 2014) (proposed mission)	Framing	Concentric, target-fixed ring tours inspired by lawn mower and milling approximation algorithms.

Pushbroom: 1D array of pixels



Pushbroom AEOS Track Selection and Scheduling problem (Lemâitre et al. 2002)

Framing: 2D array of pixels

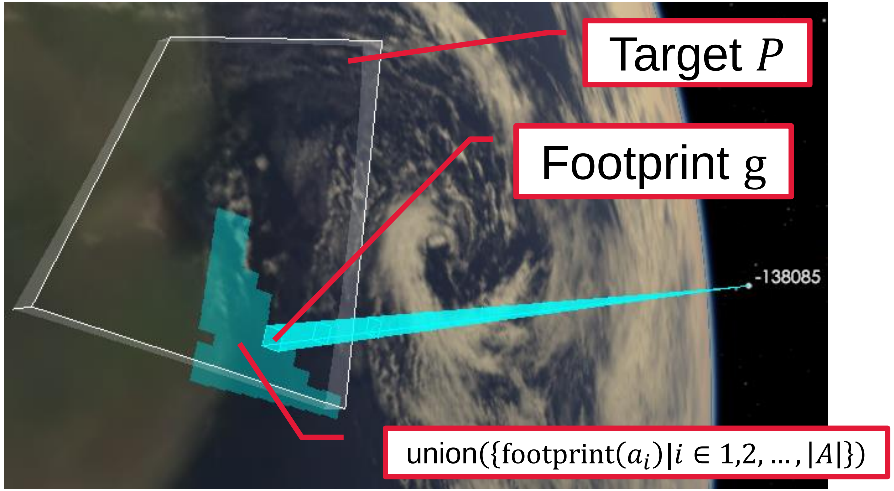


Step-stare tiling concept

Problem Statement

Given:

- A set of polygons P on the target body
$$P = \{(p_1, p_2, p_3)_i\}$$
- The set B of all possible valid observations b within horizon $[t_0, t_f]$
$$\forall\left(b = \{\mathbf{r}_{\text{tgt}}, \theta, t\}\right) \in B, t_0 < t < t_f$$
- Function to create a footprint polygon g from b
$$g \leftarrow \text{footprint}\left(\mathbf{r}_{\text{tgt}}, \theta, t\right)$$
- A Boolean valued function to check if a slew between b_i and b_j is valid
$$Boolean \leftarrow \text{slewOk}\left(b_i, b_j\right)$$



Some tour $A \subseteq B$ is valid iff
$$P \subseteq \text{union}(\{\text{footprint}(a_i) | i \in 1,2, \dots, |A|\})$$
and
$$\bigwedge_{i=1}^{|A|-1} \text{slewOk}(a_i, a_{i+1})$$

Optimization Formulation

A valid schedule A with the shortest makespan $|m|$

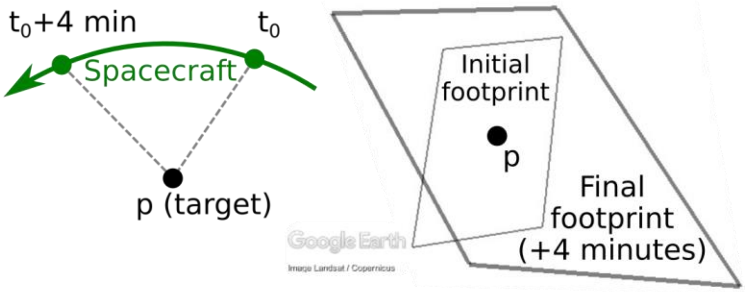
Constraints

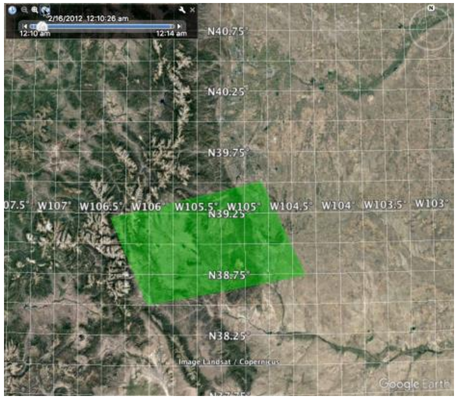
- Finite planning horizon scoped to geometric visibility
- Minimum observation duration $\Delta t_{\text{obs}} > 0$

Challenging Aspects of the Problem

Shortest schedule $|m| = \text{makespan}(A)$ optimization

- Shortest makespan optimization problem is NP-complete
- Transition (slew) cost between two target points $\mathbf{r}_{\text{tgt},i}, \mathbf{r}_{\text{tgt},i+1}$ is time varying and asymmetric (Lewellen et al. 2017)
- Shape and size of footprint change rapidly





Approximation Algorithms for Optimal Makespan

Sidewinder: Target-fixed Boustrophedon

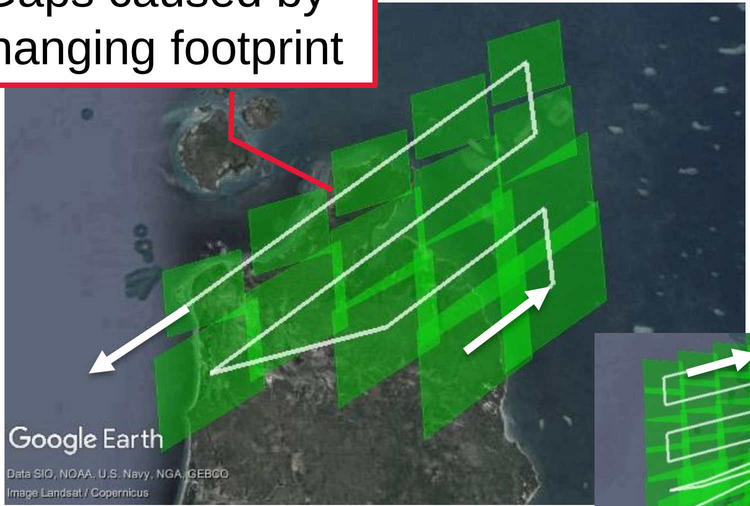
Algorithm 2 Sidewinder

> **while** $P \neq \emptyset$ **do**
> $Tour \leftarrow \textsc{planSidewinderTour}(P, \gamma, t)$
>   **while** $Tour \neq \emptyset$ **do**
> $a_i \leftarrow \textsc{pop}(Tour, t)$
>   append a_i to A
> $g \leftarrow \textsc{footprint}(a_i)$
> $P \leftarrow P - g$
> $t \leftarrow t + \Delta t_{\text{obs}} + \textsc{slewDur}(t, a_{i-1}, a_i)$
>   **end while**
> **end while**

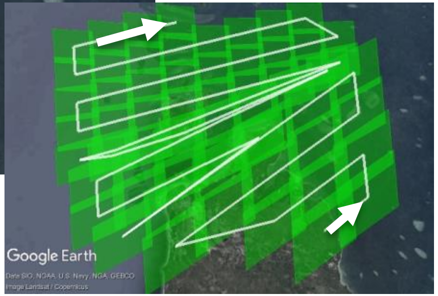
planSidewinderTour summary:

- Discretize target to a rectangular grid of rows r and columns c
- Find closest side of grid
- for $r \in r_{nearest} \dots r_{farthest}$
  - for $c \in c_{nearest} \dots c_{farthest}$
    - Tour.append(r, c)
  - Alternate column direction

Gaps caused by changing footprint



Complexity: $|A|$



Final schedule (multiple invocations)

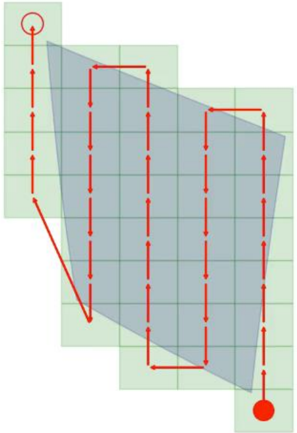
Approximation Algorithms for Optimal Makespan

Online Frontier Repair: propagate and repair a Sidewinder plan

Algorithm 5 Online Frontier Repair

Plan $Tour$
while $P \neq \emptyset$ **do**
  $\textsc{updateGrid}(Tour, F, N, X)$
  $\textsc{remove}(Tour, x \in X)$ ▷ tiles we no longer need
  $\textsc{insertCheapest}(Tour, n \in N)$ ▷ New tiles
  $a_i \leftarrow \textsc{pop}(Tour, t)$
  append a_i to A
  $g \leftarrow \textsc{footprint}(a_i)$
  $P \leftarrow P - g$
  $t \leftarrow t + \Delta t_{\text{obs}} + \textsc{slewDur}(t, a_{i-1}, a_i)$
end while

Planner's perspective



Repairs

Complexity: $|A|^2$
(looped updateGrid call)

Video and picture are from different test cases

Approximation Algorithms for Optimal Makespan

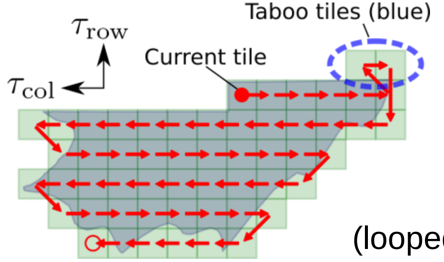
Replanning Sidewinder: replan the whole tour every step

Algorithm 3 Replanning Sidewinder

while $P \neq \emptyset$ **do**
- $\gamma_{i-1} \leftarrow \text{pop}(Tour)$ or $\text{center}(P)$ if $Tour = \emptyset$
- $\gamma \leftarrow \text{OPTIMIZEGRIDORIGIN}(\gamma_{(i-1)})$
- $Tour \leftarrow \text{PLANSIDEWINDERTOUR}(P, \gamma, t)$
- $a_i \leftarrow \text{POP}(T, t)$
- append a_i to A
- $g \leftarrow \text{FOOTPRINT}(a_i)$
- $P \leftarrow P - g$
- $t \leftarrow t + \Delta t_{\text{obs}} + \text{SLEWDUR}(t, a_{i-1}, a_i)$

end while

Locally optimize the next grid for maximum forward progress s.t. the plan contains no taboo tiles



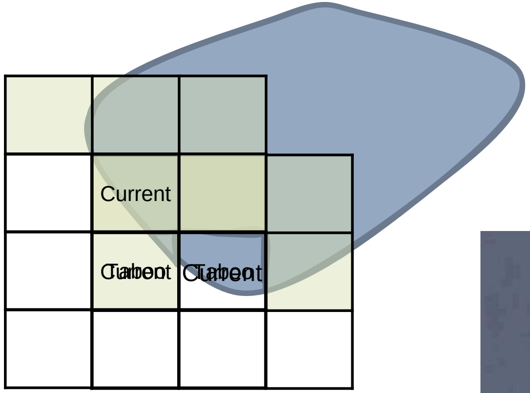
Complexity: $|A|^2$
(looped planSidewinderTour call)

Approximation Algorithms for Optimal Makespan

Grid Nibbler: local planning guided by heuristic scoring

Algorithm 8 Nibbler

while $P \neq \emptyset$ **do**
  $best \leftarrow \textsc{CheckNeighbors}(prev)$
  if $\textsc{Area}(best, t) < \epsilon$ **then**
    $newStart \leftarrow closesttargetcorner$
    $best \leftarrow \textsc{CheckNeighbors}(newStart, t)$
    if $\textsc{Score}(newStart, t) > \textsc{Score}(best, t)$ **then**
      $best \leftarrow newStart$
    end if
  end if
  $a \leftarrow \textsc{MakeObservation}(best, t)$
  append a_i to A
  $g \leftarrow \textsc{Footprint}(a_i)$
  $P \leftarrow P - g$
  $t \leftarrow t + \Delta t_{\text{obs}} + \textsc{SlewDur}(t, a_{i-1}, a_i)$
end while



score() is a heuristic function for progress toward the goal state $P = \emptyset$. Examples:

- Target area satisfied
- How closely the move resembles a human expert strategy (i.e. follow perimeter)

Complexity: $|A|$

Heuristic: radial distance from center

Experiment Methodology

Computer: 2.6 GHz, 16 GiB RAM MacBook Pro

Experiment 1
Impact of observer agility

- Fix the observer/target: difficult observer, easy target
- Vary agility (time to complete a slew), measure makespan of resulting schedule

	CICLOP	THEIA
Horizontal FOV	5.73º	1º
Vertical FOV	4.26º	1º
Image duration	0.17s	1s

Experiment 2
Algorithm Comparison

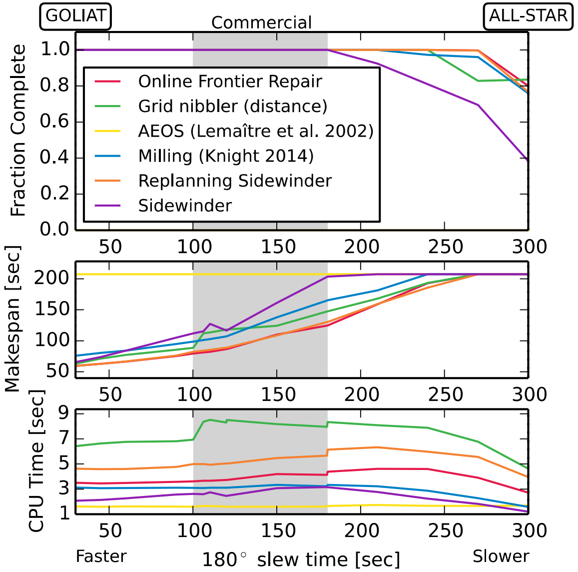
- 4 test cases: cross-product of observer capability and target difficulty

		Easy	Hard
Observer	Agility	GOLIAT	Commercial
	Imager	CICLOP	THEIA
	Orbit Altitude (km)	309×1441	615
Target	Area (km^2)	226381	8181

Results

Experiment 1: Impact of Observer Agility

- More agile: algorithm choice doesn't matter
- Less agile: algorithm choice matters
- CPU runtime increases as observer agility decreases, until 250sec/180º, where algorithms start failing to complete the target



Results

Experiment 2: Algorithm Comparison

		Easy Observer (GOLIAT)					Hard Observer (Hybrid)				
	Algorithm	CPU	RAM	$\|m\|$	$\|A\|$	%	CPU	RAM	$\|m\|$	$\|A\|$	%
Easy target	Online Frontier Repair	2s	0.04MB	1s	1	100	4s	3.14MB	87s	54	100
Easy target	Replanning Sidewinder	2s	0.08MB	2s	2	100	5s	2.11MB	89s	54	100
Easy target	Milling (Knight 2014)	11s	0.04MB	11s	8	100	3s	0.37MB	107s	64	100
Easy target	Sidewinder	2s	0.05MB	2s	2	100	2s	0.30MB	117s	63	100
Easy target	Grid Nibbler (distance)	4s	0.05MB	1s	1	100	8s	4.13MB	118s	72	100
Easy target	Grid Nibbler (area)	3s	0.05MB	1s	1	100	14s	3.71MB	109s	52	100
Hard target	Online Frontier Repair	7s	3.21MB	87s	48	100	80s	22.80MB	39429s	387	32
Hard target	Replanning Sidewinder	9s	2.19MB	81s	41	100	-	-	-	-	-
Hard target	Milling (Knight 2014)	6s	0.42MB	118s	68	100	24s	3.80MB	39430s	343	30
Hard target	Sidewinder	3s	0.22MB	74s	43	100	19s	2.30MB	39430s	389	18
Hard target	Grid Nibbler (distance)	19s	4.52MB	96s	52	100	56s	23.20MB	39428s	391	34
Hard target	Grid Nibbler (area)	20s	3.73MB	70s	39	100	146s	23.30MB	39429s	392	41

Hard/Hard case inadmissible: no valid schedules (<100% complete)

Aspect	Best algorithm	Why
Makespan	Tie: Grid Nibbler (area), Online Frontier Repair	Smallest $\|m\|$, 2/3 cases
CPU use	Sidewinder	3/3 cases
RAM use	Sidewinder	2/3 cases

Discussion

- Number of images $|A|$ is not necessarily proportional to schedule makespan $|m|$
  - Path quality (slew cost) also affects $|m|$
- Algorithm complexity not very important
  - $|A|$ Grid Nibbler requires more CPU time than $|A|^2$ algorithms
- Grid nibbler is susceptible to dead ends

Discussion

Back-of-the-Envelope: On-board CubeSat Feasibility

- Marginally feasible to execute on a Raspberry Pi compute Module 3
- Infeasible for most CubeSats
- Methodology
  - Algorithm: Sidewinder
  - Test case: Easy observer, hard target
  - Linearly scale runtime from 2.6 GHz experiment CPU to CubeSat clock rates
- Caveats
  - Ignoring CPU cache, disk I/O rate
  - Ignoring 470 MB of non-algorithm RAM overhead in our prototype (we wrote inefficient code)
  - Ignoring competing processes

	Raspberry Pi compute module 3	Vorago VA10820 (rad hard ARM)
CubeSat Mission	AAReST (Ramaprakash 2017)	DemoSat-2 (Astranis)
CPU clock	1.2 GHz	50 MHz
RAM available	2 GiB	128 KiB
Schedule rate $\frac{\lvert m \rvert}{\text{CPU time}}$	11.4x real time	0.5x real time
% RAM used	0.01%	180%
Feasible?	**Yes**	**No**

Conclusions

- Online Frontier Repair and Replanning Sidewinder algorithms outperformed the previous state of the art (Knight 2014) in all admissible test cases
- Committing the tour to the target body early gives poor results
- Choice of algorithm matters most when the observer is marginally capable of satisfying its target
- No clear best algorithm: portfolio approach may work best

Recommendations for Future Work

- An actual satellite should fly one of these algorithms
- Higher fidelity spacecraft agility models
- Apply backtracking, beam search and other traditional grid search techniques to grid nibbler
- Broader comparison of algorithms across more problem instances

Jet Propulsion Laboratory
California Institute of Technology

jpl.nasa.gov